SCHEDULING THE DISPATCH OF CELLS IN MULTISTAGE SWITCHES

§ 0. PRIORITY CLAIM

Benefit is claimed, under 35 U.S.C. § 119(e)(1), to the filing date of: provisional patent application serial number 60/252,006, entitled "CRRD: A CONCURRENT ROUND-ROBIN DISPATCHING SCHEME FOR CLOS-NETWORK SWITCHES", filed on November 20, 2000 and listing Jonathan Chao and Eiji Oki as the inventors; and provisional patent application serial number 60/253,335, entitled "A SCALABLE ROUND-ROBIN BASED DISPATCHING SCHEME FOR LARGE-SCALE CLOS-NETWORK SWITCHES", filed on November 27, 2000 and listing Jonathan Chao and Eiji Oki as inventors, for any inventions disclosed in the manner provided by 35 U.S.C. § 112, ¶ 1. This provisional application is expressly incorporated herein by reference.

§ 1. BACKGROUND

§ 1.1 FIELD OF THE INVENTION

The present invention concerns the communication of data over networks, such as the Internet for example. More specifically, the present invention concerns scheduling the servicing (e.g., dispatching) of cells or packets buffered at input ports of a switch.

§ 1.2 RELATED ART

Switches and routers are used in networks, such as the Internet for example, to forward data towards its destination.

Express Mail No. EL667251302US

The need for large-scale switches and routers is introduced in § 1.2.1 below. Then, two types of switches, as well as disadvantages and challenges in each of these two types of switches, are introduced in § 1.2.2 below.

§ 1.2.1 THE NEED FOR LARGE-SCALE (e.g., TERABIT) ROUTERS AND SWITCHES

Many expect that Internet traffic will continue to grow explosively. Given this assumption, high-speed switches and routers (e.g., those having a throughput over one Terabit per second) will become necessary. Most high-speed packet switches adopt a fixed-size cell in the switch fabric. If variable length packets are to be supported in the network, such packets may be segmented and/or padded into fixed-sized cells upon arrival, switched through the fabric of the switch, and reassembled into packets before departure. There are two main types of high-speed switches, each introduced in § 1.2.2 below.

§ 1.2.2 TYPES OF SWITCHES (SINGLE STAGE VERSUS MULTI-STAGE) AND THEIR CHARACTERISTICS

There are two main types of high-speed switches -- namely single stage and multi-stage. Single stage switches and perceived limits of single stage switches are introduced in §§ 1.2.2.1 and 1.2.2.2, respectively, below. Similarly, multi-stage switches and perceived limits of multi-stage switches are introduced in §§ 1.2.2.3 and 1.2.2.4, respectively, below.

§ 1.2.2.1 SINGLE STAGE SWITCH ARCHITECTURES

A so-called "crossbar" switch is a classic example of a single stage switch. In a crossbar switch, identical switching elements are arranged on a matrix plane. The article, N. McKeown, M. Izzard, A. Mekkiltikul, W. Ellerisick and M. Horowitz, "Tiny-Tera: A Packet Switch Core," IEEE Micro., pp. 26-33, (Jan.-Feb. 1997) (incorporated herein by reference and hereafter referred to as "the Tiny-Tera article") proposes a 320 gigabit per second crossbar switch fabric that uses a centralized scheduler referred to as "iSLIP". The article H. J. Chao and J-S Park, "Centralized Contention Resolution Schemes for a Large-Capacity Optical ATM Switch," Proc. IEEE ATM Workshop '97 (Fairfax, VA, May 1998) (incorporated herein by reference and hereafter referred to as "the Chao article") introduced a centralized contention resolution scheme for a large capacity crossbar optical switch. The article E. Oki, N. Yamanaka, Y. Ohtomo, K. Okazaki and R. Kawano, "A 10-Gb/s (1.25 Gb/s x 8) 4 x 2 0.25 μm CMOS/SIMOX ATM Switch Based on Scalable Distributed Arbitration," IEEE J. of Solid-State Circuits, Vol. 34, No. 12, pp. 1921-1934 (Dec. 1999) (incorporated herein by reference and hereafter referred to as "the Oki article") describes a high-speed switch chip having a throughput of 40 Gb/s, for a high-speed crosspoint-buffered crossbar switching system.

§ 1.2.2.2 LIMITS OF SINGLE STAGE SWITCH ARCHITECTURES

The switching techniques used in single stage switches, such as those described in the articles referenced in § 1.2.2.1, are effective up to a certain switch size.

Unfortunately, however, with such techniques, the complexity of the switching elements is proportional to the square of the number of switch ports. As a practical matter, this limits the feasibility of using such techniques in a large scale (e.g., Terabit per second and above) switch.

§ 1.2.2.3 MULTI-STAGE SWITCH ARCHITECTURES

In view of the limits of single stage switches, multiple-stage switch architectures have been introduced. For example, the so called "Clos-network" switch has three stages and is very attractive because of its scalability. See the article, C. Clos, "A Study of Non-Blocking Switching Networks," Bell Sys. Tech. Jour., pp. 406-424 (March 1953) (incorporated herein by reference and hereafter referred to as "the Clos article"). More specifically, the three stages include (i) input modules, (ii) a central switching fabric (incorporated in central modules), and (iii) output modules.

Clos-network switches have been categorized into two types -- those with buffers to store cells in the second-stage (central) modules and those with no buffers in the second-stage (central) modules. For example, the article, T. Chaney, J. A. Fingerhut, M. Flucke, J. S. Turner, "Design of a Gigabit ATM Switch," Proc. IEEE INFOCOM '97, pp. 2-11 (April 1997) (incorporated herein by reference and hereafter referred to as "the Chaney article") discusses a gigabit ATM switch using buffers in the second-stage modules. In the switch architecture discussed in the Chaney article (hereafter referred to as "the Chaney switch"), every incoming cell is randomly distributed from the first-stage to the second-stage module to balance the traffic load in the second-stage. This is commonly referred to

as "random dispatching". The buffers in the second-stage modules are used to resolve contention among cells from different first-stage modules. See the article J. Turner and N. Yamanaka, "Architectural Choices in Large Scale ATM Switches," IEICE Trans. Commun., Vol. E81-B, No. 2, pp. 120-137 (Feb. 1998) (incorporated herein by reference and hereafter referred to as "the Turner article").

An example of a switch architecture with no buffers in the second-stage modules is provided in the article F. M. Chiussi, J. G. Kneuer, and V. P. Kumar, "Low-Cost Scalable Switching Solutions for Broadband Networking: The ATLANTA Architecture and Chipset," IEEE Commun. Mag., pp. 44-53 (Dec. 1997) (incorporated herein by reference and hereafter referred to as "the Chiussi article"). Since there are no buffers in the second-stage modules to resolve potential contention, how cells are dispatched from the first-stage to the second-stage becomes important. The simple random distribution used in the Chaney switch may be used in the switch discussed in the Chiussi article (hereafter referred to as "the Chiussi switch") under certain conditions. Given the potential for contention, some buffers are provided in the first and third stages of the Chiussi switch. Although there are some studies of routing algorithms where every stage has no buffers, such algorithms require a contention resolution function for output ports, before cells even enter the multiple-stage switches. See, e.g., the articles, C. Y. Lee and A. Y. Qruc, "A Fast Parallel Algorithm for Routing Unicast Assignments in Benes Networks," IEEE Trans. on Parallel and Distributed Sys., Vol. 6, No. 3, pp. 329-333 (March 1995), and T. T. Lee and S-Y Liew, "Parallel Routing Algorithms in Benes-Clos Networks," Proc. IEEE INFOCOM '96, pp. 279-286 (1996). (Both of these articles are

incorporated herein by reference.) Such a pre-switch contention resolution function is challenging to implement in high-speed switches.

§ 1.2.2.4 PROBLEMS WITH KNOWN MULTIPLE-STAGE SWITCH ARCHITECTURES

To provide high performance (e.g., high throughput), known multiple-stage switch architectures will typically require speed-up of the internal switching fabric (i.e., in the second-stage modules) and/or resequencing. For example, the Chaney switch provides high performance if the internal speed-up factor is set to be more than 1.25. As is known, speed-up of the switching fabric can be accomplished by increasing the number of central modules and/or increasing the internal link speed used in the central modules. Unfortunately, however, in the Chaney switch, buffers used in the second-stage modules cause cells to become out-of-sequence. Since the Chiussi switch does not use buffers in its second-stage modules, cell out-of-sequence problems are not encountered. Assuming that a random dispatching scheme is used to forward cells from the first stage to the second stage of the Chiussi switch, to minimize the chance of contention at the second stage, the internal speed-up factor has to be increased further. For example, to achieve 100 % throughput using random dispatching in a large-size Chiussi switch, the speed-up is set to about 1.6 as indicated in the Chiussi article.

As should be appreciated from the foregoing, to be used in today's more demanding environments, the Chaney switch requires cell resequencing, which is challenging, and at least moderate speed-up, which increases costs. To be used in today's

more demanding environments, the Chiussi switch requires even more speedup, which increases costs even more. In view of these problems with known switch architectures, a scalable switch architecture having a high throughput, avoiding out-of-sequence cells (e.g., by eliminating buffers in the second stage), and avoiding speed-up of the switch fabric is coveted.

§ 2. SUMMARY OF THE INVENTION

The present invention may be used to provide a scalable switch architecture that has a high throughput, avoids out-of-sequence cells, and avoids speedup of the switch fabric is coveted. The present invention may do so by providing a cell dispatch scheduling method for use in a multi-stage switch including a plurality of central modules, and a plurality of input modules, each including virtual output queues and outgoing links coupled with each of the plurality of central modules. In one embodiment of the cell dispatch scheduling method, a non-empty virtual output queue of an input module is matched with an outgoing link in the input module, and the outgoing link of the input module is matched with an outgoing link of one of the central modules.

In one embodiment, a non-empty virtual output queue of an input module may be matched with an outgoing link in the input module by (i) broadcasting a request for the non-empty virtual output queue to an arbiter for each of the outgoing links of the input module, (ii) selecting, with the arbiter of each of the outgoing links of the input module, a non-empty virtual output queue that broadcast a request, (iii) sending a grant to an arbiter for the

selected non-empty virtual output queue, and (iv) selecting, with the arbiter of the selected non-empty virtual output queue, an outgoing link from among the one or more outgoing links that sent a grant. In one version of such an embodiment, the act of matching a non-empty virtual output queue of an input module with an outgoing link in the input module may occur within one cell time slot. In one version of such an embodiment, the act of selecting, with the arbiter of each of the outgoing links of the input module, a non-empty virtual output queue that broadcast a request, may be done in accordance with a round robin discipline. The round robin discipline may be effected by a pointer. In one embodiment, the pointer may move through groups of virtual output queues, before moving through virtual output queues within each group.

In one embodiment, the acts of (i) broadcasting a request for the non-empty virtual output queue to an arbiter for each of the outgoing links of the input module, (ii) selecting, with the arbiter of each of the outgoing links of the input module, a non-empty virtual output queue that broadcast a request, (iii) sending a grant to an arbiter for the selected non-empty virtual output queue, and (iv) selecting, with the arbiter of the selected non-empty virtual output queue, an outgoing link from among the one or more outgoing links that sent a grant, may be performed at least twice within one cell time slot.

In one embodiment, the act of matching the outgoing link of the input module with an outgoing link of one of the central modules may be performed by (i) broadcasting a request for the outgoing link of the input

module to an arbiter for each of the outgoing links of the central modules that lead towards an output port associated with the virtual output queue matched with the outgoing link of the input module, (ii) selecting with the arbiter of each of the outgoing links of the central modules, an outgoing link of an input module that broadcast a request; and (iii) sending a grant to the selected outgoing link of the input module.

The teachings of the present invention may be used for matching a non-empty virtual output queue of an input module with an outgoing link in the input module.

The present invention also provides a teaching of apparatus for effecting the various methods. The present invention also provides a teaching of various data structures that may be used for effecting the various methods.

§ 3. BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of an exemplary multiple-stage switch in which the present invention may be implemented.

Figure 2 is a bubble diagram of operations that may be performed by the present invention.

Figure 3 is a flow diagram illustrating an exemplary virtual output queue method that may be performed under the present invention.

Figure 4 is a flow diagram illustrating an exemplary link arbitration method that may be performed under the present invention.

Figure 5 is a flow diagram illustrating an exemplary virtual output queue arbitration method that may be performed under the present invention.

Figure 6 is a flow diagram illustrating an exemplary outgoing link method that may be performed under the present invention.

Figure 7 is a flow diagram illustrating an exemplary central module arbitration method that may be performed under the present invention.

Figure 8 is a block diagram of components of an exemplary apparatus, as well as inter-component signaling links, that may be used to perform operations under the present invention.

Figures 9(a) through 9(g) illustrate an example of how a first phase of an exemplary dispatching method under the present invention matches non-empty virtual output queues to outgoing links in an input module.

Figure 10 illustrates exemplary data structures that may be used to store state information for use by a link arbitration operation.

Figure 11 illustrates exemplary data structures that may be used to store state information for use by a

VOQ arbitration operation.

Figure 12 illustrates exemplary data structures that may be used to store state information for use by a central module arbitration operation.

§ 4. DETAILED DESCRIPTION OF THE INVENTION

The present invention involves novel methods, apparatus and data structures for dispatching cells or packets buffered at the input port of a multi-stage switch. The following description is presented to enable one skilled in the art to make and use the invention, and is provided in the context of particular embodiments and methods. Various modifications to the disclosed embodiments and methods will be apparent to those skilled in the art, and the general principles set forth below may be applied to other embodiments, methods and applications. Thus, the present invention is not intended to be limited to the embodiments and methods shown and the inventors regard their invention as the following disclosed methods, apparatus and materials and any other patentable subject matter to the extent that they are patentable.

§ 4.1 EXEMPLARY ENVIRONMENT

The present invention may be practiced in a multiple-stage switch, such as a Clos-network switch for example. Figure 1 is a high-level block diagram of a Clos-network switch 100. The switch 100 basically includes input modules ("IMs") 120 at a first stage, central modules ("CMs") 140 at a second stage, and output modules ("OMs")

160 at a third stage.

As shown in Figure 1, each input module (IM) 120 includes a number (n) of input ports ("IPs") 110. Thus, if there are a number (k) of input modules (IMs) 120, there will be a total of n*k input ports (IPs) 110. Similarly, each output module (OM) 160 includes a number (n) of output ports ("OPs") 170. Thus, if there are a number (k) of output modules (OMs) 160, there will be a total of n*k output ports (OPs) 170. Each output port (OP) 170 may include a buffer 165 receiving, at most, m cells in one cell time slot. The buffer size should be large enough to avoid cell loss. Each output port (OP) 170 forwards one cell in a first-in-first-out (FIFO) manner, to an associated output line.

A number (m) of central modules (CMs) 140 are arranged between the input modules (IMs) 120 and output modules (OMs) 160. More specifically, as shown, each input module (IM) 120 includes a number (m) of outgoing links L_i 130, each connecting the input module (IM) 120 to a different one of the m central modules (CMs) 140. Similarly, each central module (CM) 140 includes a number (k) of outgoing links L_o 150, each connecting the central module (CM) 140 with a different one of the k output modules (OMs) 160.

Finally, each of the input modules (IM) 120 may include a number of virtual output queues (VOQs) 125, each of the VOQs 125 being associated with an output port (OP) 170. The VOQs 125 are used to eliminate problems caused by head-of-line ("HOL") blocking. A given VOQ 125 can receive

at most n cells from n input ports (IPs) 110, and can send one cell to a central module (CM) 140 in one cell time slot.

The following terminology:

$n \equiv$ the number of input ports and output ports for each IM and OM, respectively;

$k \equiv$ the number of IMs, as well as the number of OMs;

$m \equiv$ the number of CMs;

$IM(i) \equiv$ the ith input module, where $0 \le i \le k-1$;

$CM(r) \equiv$ the rth central module, where $0 \le r \le m-1$;

$OM(j) \equiv$ the jth output module, where $0 \le j \le k-1$;

$IP(i,h) \equiv$ the hth input port at IM(i), where $0 \le h \le n-1$;

$OP(j,h) \equiv$ the h^{th} output port at OM(j), where $0 \le h \le n-1$;

$VOQ(i,j,h) \equiv$ the VOQ in IM(i) that stores cells destined for OP(j,h);

$L_i(i,r) \equiv$ the link between IM(i) and CM(r); and

$L_o(r,j) \equiv$ the link between CM(r) and OM(j),

may be used in the specification that follows.

The first stage of the switch 100 may include k input modules (IMs) 120, each of which has an n-by-m dimension. The second stage of the switch 100 may include m central modules (CMs) 140, each of which has a k-by-k dimension. The central modules (CMs) 140 are preferably buffer-less, thereby avoiding the cell out-of-sequence problems introduced in § 1.2.2.4 above. The third stage of the switch 100 may include k output modules (OMs) 160, each

of which has an m-by-n dimension.

§ 4.2 FUNCTIONS THAT MAY BE PERFORMED

A first aspect of the present invention may function to provide a scalable multiple-stage switch, able to operate at high throughput, without needing to resort to speeding up the switching fabric and without needing to use buffers in the second stage. Accordingly, the present invention may avoid the cost of speed-up and the cell out-of-sequence problems that may occur when buffers are used in the second stage. The present invention may do so using a multiple phase cell dispatch scheme, each phase using a simple and fair (e.g., round robin) arbitration methods. More specifically, the present invention may function to provide a multiple phase cell dispatch scheme in which VOQs of an input module and outgoing links of the input module are matched in a first phase, and in which an outgoing link of an input module is matched with an outgoing link of a central module in a second phase. The arbiters become desynchronized under stable conditions which contributes to the switch's high throughput characteristic.

The present invention may also function to relax a dispatch scheduling time and reduce the complexity of interconnections between arbiters. The present invention may do so by arranging output link arbiters as master and slave arbiters, operated in a hierarchical manner. More specifically, the VOQs of an input module may be arranged into groups. For each outgoing link of the input module, a master arbiter may select a group of VOQs from among a

number of candidate groups, and a slave arbiter may select a VOQ from among the VOQs belonging to the selected group.

Finally, the present invention may function to relax the time (e.g., from less than one cell time slot to more than one cell time slot) needed to schedule a cell dispatch. The present invention may do so by introducing more than one subscheduler, each of which is allowed to take more than one time slot for dispatching, although one of the subschedulers provides a dispatching result within each cell time slot.

§ 4.3 EXEMPLARY OPERATIONS

Figure 2 is a bubble chart illustrating operations that may be performed in accordance with the present invention. When a cell 205 arrives at a VOQ, a VOQ (non-arbitration) operation 210 may broadcast (as indicated by ellipses) a request 215 to a plurality of link arbitration operations 230, each associated with a given outgoing link 130 of the input module (IM) 120. The request 215 is indicated as a first communication ("1").

Based on link arbitration state information (e.g. a round-robin pointer) 235, in each case, the link arbitration operations 230 will select one of the candidate VOQs that submitted a request 215. It 230 will then send a grant 237 to an arbitration operation 240 of the selected VOQ. The grant 237 is indicated as a second communication ("2"). The link arbitration operation 230 may also send declines (not shown) to non-selected VOQs, though such a communication(s) is not necessary.

Since the VOQ operation 210 broadcasts a request (e.g., to all the outgoing links of the input module), it's corresponding arbitration operation 240 may receive more than one grant (i.e., one grant each from more than one outgoing link arbitration operation 230). Accordingly, the VOQ arbitration operation 240 uses VOQ arbitration state information (e.g., a round robin pointer) 245 to select one of the outgoing links from those candidates providing a grant. The VOQ arbitration operation 240 then communicates a grant 247 to link operations 250 associated with the selected link. This grant is indicated as third communication ("3").

At this point, a cell at a VOQ may have been (i.e., if it won an arbitration) matched with an outgoing link 130 of the input module (IM) 120. These operations related to generating such a VOQ-L_i match may be referred to as the first phase of the cell dispatch scheduling invention. This first phase can be summarized as two steps. In the first step, at each time slot, non-empty VOQs send (e.g., multicast, or preferably broadcast) requests to use any one of a number of outgoing links L_i (of an input module (IM)) in a next cell time slot. In the second step, each outgoing link L_i sends a grant to a selected VOQ. If a given VOQ receives more than one grant, it may select one of the outgoing links from which it received a grant. Accordingly, the input module (IM) selects at most m request out of n*k non-empty VOQs. If these steps can be repeated within a cell time slot, multiple iterations can be run such that non-empty VOQs failing to be matched with an available outgoing link may

retry during a next iteration. If multiple iterations are run, the VOQs and/or outgoing links L_i may maintain state information related to whether they are available or reserved for the next cell time slot.

Still referring to Figure 2, a link operation 250 associated with each outgoing link 130 matched with a VOQ, may submit a request 255 to a central module (CM) arbitration operation 260. The central module (CM) is associated with the outgoing link (L_i). Each central module will have k arbiters, each corresponding to an output module (OM). The arbiter serving the request will correspond with the output module (OM) defined by the non-empty VOQ that is matched with the outgoing link L_i that originated the request. This request 255 is indicated as a fourth communication ("4"). Alternatively, the VOQ selecting an outgoing link can originate such a request (4') as indicated by dashed line 290.

Each CM arbitration operation 260 may select one of the candidate requesting outgoing links 130 based on IM-CM arbitration state information 265. It 260 then communications a grant 267 back to the link operation 250 associated with the selected one of the candidate requesting output links 130. This grant 267 is indicated as a fifth communication ("5").

The link operation 250 may then send an indication 270 to the VOQ operation 210 that it may send (i.e., dispatch) its head-of-line ("HOL") cell. This indication 270 is indicated as a sixth communication ("6"). The various arbitration operations 230, 240, 260 may update

their state information 235, 245, 265. Further, other state information 212 may be updated by its associated operation 210.

Various alternatives of when state information is updated will be apparent to those skilled in the art. For example, each of the outgoing link arbitration operation(s) 230 may employ a pointer (to VOQ identifiers), updated in a round-robin manner. The pointer may be updated (a) when a VOQ request is granted, (b) when a VOQ grant is received, or, preferably (c) when a central module grant is received. In servicing VOQs, it is possible for the arbitration operation 230 to pass over empty (i.e., non-requesting) VOQs. Therefore, when the pointer is updated, it can (a) go to a next VOQ from the VOQ pointed to at the start of the cell time slot, or (b) go to a next VOQ from the VOQ selected.

Similarly, each of the VOQ arbitration operation(s) 240 may employ a pointer (to outgoing links), updated in a round-robin manner. The pointer may be updated (a) when a grant is received from an outgoing link L_i, (b) when an outgoing link is selected by the VOQ arbitration operation, or (c) when a central module grant is indicated. It is possible for the arbitration operation 240 to pass over non-granting outgoing links L_i. Therefore, when the pointer is updated, it can (a) go to a next outgoing link L_i from the outgoing link L_i pointed to at the start of the cell time slot, or (b) go to a next outgoing link L_i from the selected outgoing link L_i.

Finally, each of the CM arbitration operation(s) 260 may employ a pointer (to outgoing links L_i), updated in a round-robin manner. The pointer may be updated upon a grant by the CM arbitration operation 260. In servicing outgoing links, it is possible for the CM arbitration operation to 260 to pass over non-requesting outgoing links L_i. Therefore, when the pointer is updated, it can (a) go to a next outgoing link L_i from the outgoing link L_i pointed to at the start of the cell time slot, or (b) go to a next outgoing link L_i from the selected outgoing link L_i.

Having described various operations that may be performed in accordance with the present invention, exemplary apparatus, methods and data structures for implementing such operations are now described in § 4.4 below.

§ 4.4 EXEMPLARY METHODS, DATA STRUCTURES, AND APPARATUS FOR PERFORMING THE EXEMPLARY OPERATIONS

Exemplary methods and data structures for implementing various operations of the present invention are described in § 4.4.1. Then, exemplary apparatus for implementing various operations of the present invention are described in § 4.4.2.

§ 4.4.1 EXEMPLARY METHODS AND DATA STRUCTURES

Figure 3 is a high-level flow diagram of an exemplary method 210′ for effecting various VOQ operations 210. Referring to conditional branch point 310 and block 320, if a cell has arrived at the VOQ but has not yet won

arbitration (the VOQ is "non-empty"), a request is broadcast to all link arbiters (or link arbitration operations 230) of the input module (IM) 120 to which the VOQ belongs. (Recall, e.g., the first communication 215 of Figure 2.)

Still referring to Figure 3, as indicated by conditional branch point 330 and block 340, if the cell (or the VOQ in general) won arbitration (e.g., all rounds of arbitration -- through to the central module 140) (Recall, e.g., the sixth communication 270 of Figure 2.), the cell is sent (e.g., in the upcoming cell time slot). Although not shown in Figure 3, these steps may be continuously run. State information 212 of the VOQ may be updated at this point. Alternatively, blocks 320 and 340 may be triggered upon the occurrence of the relevant events.

Figure 4 is a high-level flow diagram of an exemplary method 230′ for effecting a link arbitration operation 230. Referring to conditional branch point 410, it is determined whether or not a request or requests were received from a VOQ or VOQs. If not, the method 230′ is left via RETURN node 440. If, on the other hand, such a request or requests is received, the rest of the method 230′ is effected. Thus, the receipt of request(s) from VOQ(s) may serve to trigger the main steps of the method 230′.

As indicated by block 420, the link arbitration method selects one VOQ from among those sending requests. (Recall, e.g., 230 and 235 of Figure 2.) The arbitration 420 may be done based on the position of a pointer which is

updated in accordance with a round-robin discipline. The pointer may move through all of the VOQs. If the pointer currently points to a VOQ that did not send a request, it may move to a next VOQ, repeatedly, until it points to a VOQ that submitted a request. Referring back to Figure 4, the method 230′ then sends a grant to the selected VOQ as indicated by block 430. (Recall, e.g., the second communication 237 of Figure 2.) The method 230′ may then be left via RETURN node 440. Note that state information (e.g., the pointer) may be updated at this point, or, alternatively, may be updated later.

Figure 10 illustrates exemplary state information 235′ that may be used by the link arbitration operation 230′. As shown, a table 1010 may include a plurality of records, each record including a VOQ identifier 1012 and a field 1014 indicating whether or not a request was received (in the current iteration) from the corresponding VOQ. A pointer 1016 may cycle through the records in a round-robin manner. The VOQ identifiers 1012 may be ordered such that a given VOQ within various groups of VOQs are serviced first, before a next VOQ within the various groups of VOQs are serviced. As will be appreciated from the examples described in § 4.5 below, such an ordering of the VOQs hastens an advantageous desynchronization of pointers.

The data structure 1020 may be used to indicate whether or not the outgoing link is reserved for a next cell time slot. If so, the link arbitration method 230′ can ignore requests from VOQs (and/or inform the VOQs that it is taken). The data structure 1030 may be used to indicate whether or not a VOQ, which was selected, chose

the outgoing link in its own arbitration. Finally, data structure 1040 may be used to indicate whether or not a central module request was granted.

Figure 5 is a high-level flow diagram of an exemplary method 240′ for effecting a VOQ arbitration operation 240. Referring to conditional branch point 510, it is determined whether or not one or more grants are received from the link arbitration operations 230. (Recall, e.g., the second communication 237 of Figure 2.) If not, depending on the cell time slot and the time needed for a dispatch determination (i.e., to match a VOQ with a CM), additional iterations (requests) may be possible. Accordingly, as indicated in conditional branch point 520 and block 530, if there is time left for another iteration, the VOQ arbitration method 240′ may inform a VOQ operation 210 to rebroadcast a request.

Referring back to conditional branch point 510, if one or more grants are received, the method continues to block 540 where one of the candidate grants is selected. The selection 540 may be done based on the position of a pointer which is updated in accordance with a round-robin discipline. The pointer may move through all of the outgoing links 130. If the pointer currently points to an outgoing link 130 that did not send a grant, it may move to a next outgoing link 130, repeatedly, until it points to an outgoing link 130 that submitted a grant. Then, as shown in block 550, a grant is sent to the outgoing link operation associated with the selected one of the candidate grants. Although not shown, in an alternative method, the VOQ can submit a request, on behalf of the selected

outgoing link (L_i), to a central module arbitration. (Recall 290 of Figure 2.) The method 240′ may then be left via RETURN node 560.

Figure 11 illustrates exemplary state information 245′ that may be used by the VOQ arbitration operation 240. As shown, table 1110 may include a plurality of records, each record including an outgoing link (L_i) identifier 1112 and a field 1114 indicating whether or not a grant was received (in the current iteration) from a corresponding outgoing link. A pointer 1116 may cycle through the records in a round-robin manner.

The data structure 1120 may be used to indicate whether or not a cell is buffered at the VOQ. The data structure 1130 may be used to indicate whether or not a grant(s) was received from an outgoing link(s). If not, no VOQ arbitration operation 240 is needed. Finally, the data structure 1140 may be used to indicate whether or not a central module grant was received (e.g., by a matched outgoing link (L_i)). Such information may be used, for example, for updating the pointer 1116.

Figure 6 is a high-level flow diagram of an exemplary method 250′ for effecting an outgoing link operation 250. Referring to conditional branch point 610, it is determined whether or not a VOQ grant is received. (Recall, e.g., the third communication 247 of Figure 2.) If so, a request is sent to a central module arbiter, as indicated by block 620. (Recall, e.g., the fourth communication 255 of Figure 2.) Recall that in one alterative, the VOQ may submit such a request on behalf of

the outgoing link (L_i).

Referring to conditional branch point 630, it is determined whether or not a grant is received from the central module arbiter. (Recall, e.g., the fifth communication 267 of Figure 2.) If so, the VOQ (e.g., VOQ operation 210) matched with the outgoing link 130 is informed (so that it can send its head-of-line (HOL) cell. (Recall, e.g., the sixth communication 270 of Figure 2.) Although not shown in Figure 6, these steps may be continuously run. Alternatively, blocks 620 and 640 may be triggered upon the occurrence of the relevant events.

Figure 7 is a high-level flow diagram of an exemplary method 260′ for effecting a central module arbitration operation 260. Referring to conditional branch point 710, it is determined whether or not one or more requests were received from one or more outgoing links (L_i) 130. If not, the method 260′ may be left via RETURN node 740. If, on the other hand, one or more requests were received from one or more outgoing links 130, the central module arbitration method 260′ may select one link from among the candidate links that sent a request, as indicated by block 720. The selection 720 may be done based on the position of a pointer which is updated in accordance with a round-robin discipline. The pointer may move through all of the outgoing links 130. If the pointer currently points to an outgoing link 130 that did not send a request, it may move to a next outgoing link 130, repeatedly, until it points to an outgoing link 130 that submitted a request. Referring to block 730, the method 260′ may then send a grant to the selected link. (Recall, e.g., the fifth

communication 267 of Figure 2.) The method 260′ may then be left via RETURN node 740.

Figure 12 illustrates exemplary state information 265′ that may be used by the central module arbitration operation 260. As shown, table 1210 may include a plurality of records, each record including an outgoing link (L_i) identifier 1212 and a field indicating whether or not a request was received from the corresponding outgoing link (L_i). A pointer 1216 may cycle through the records in a round-robin manner. The data structure 1220 may be used to indicate whether or not an outgoing link (L_i) was selected in the previous or current cell time slot.

Note that if a request from a matched VOQ-outgoing link is not granted, such a request may be resent to a central-module arbiter in a next cell time slot (e.g., if pointers related to ungranted requests are not updated.)

§ 4.4.2 EXEMPLARY APPARATUS

Figure 8 is a high-level block diagram illustrating exemplary components and interconnections of an exemplary apparatus 800 that may be used to effect at least some of the operations of the present invention. An exemplary input module (IM) 120′ may include virtual output queues (VOQs) 810, each of which includes (or more generally, is associated with a corresponding) an arbiter 815, and link controllers 820, each of which includes (or more generally, is associated with a corresponding) an arbiter 825. An exemplary central module 140′ may include

arbiters 830. The VOQ operations 210 may be effected on the virtual output queues 810. The VOQ arbitration operations 240 may be effected on the VOQ arbiters 815. The link operations 250 may be effected on the link controllers 820. The link arbitration operations may be effected on the link arbiters 825. Finally, the central module arbitration operations may be effected on the arbiters 830. Naturally, such operations may be physically located elsewhere.

Various signaling lines or links may be provided. To simplify the drawing, signaling lines coupled with VOQs 810b and 810c, or their associated arbiters 815b and 815c, are not shown. Signaling lines, depicted by solid lines 840, may be used to broadcast requests from a VOQ 810 to each of the link arbiters 825 in the input module (IM) 120'. (Recall, e.g., the first communication 215 of Figure 2.) Signaling lines, depicted by short dashed lines 850, may be used by each link controller arbiter 825 to send a grant to a selected VOQ arbiter 815. (Recall, e.g., the second communication 237 of Figure 2.) Signaling lines, depicted by dot-dash lines 860, may be used by a VOQ arbiter 815 to send a grant to a selected link controller 820. (Recall, e.g., the third communication 247 of Figure 2.) At this point, a VOQ 810, winning arbitration, may be matched with an outgoing link.

Signaling lines, depicted by double dot-dash lines 870, may be used by the link controllers 820 to send requests to the arbiters 830 of the central modules 140'. Signaling lines, depicted by double dash-dot lines 880, may be used by the central module arbiters 830 to send a grant

to a selected link controller 820. Finally, the link controllers 820 may use signaling lines 890, depicted by spaced dotted lines, to inform the appropriate VOQs 810 that they have won arbitration and can therefore send their head-of-line (HOL) cells (e.g., in the upcoming cell time slot).

The VOQs 810, their arbiters 815, the line controllers 820, their arbiters 820, and the arbiters 830 of the central modules 140′ may be effected by programmable logic arrays, application specific integrated circuits, and/or microprocessors operating in accordance with stored instructions. Memory (referred to generally as a "machine readable medium") may be used to store the various state information (Recall, e.g. elements 212, 235, 245, and 265 of Figure 2.) used by these components. Similarly, memory can be used to buffer cells at the virtual output queues 810.

§ 4.5 EXAMPLES ILLUSTRATING OPERATIONS PERFORMED BY AN EXEMPLARY EMBODIMENT

Figures 9(a) through 9(g) illustrate an example of operations of the first stage of an exemplary dispatching method. In this example, the invention is embodied in a Clos-network switch, such as that 100 illustrated in Figure 1. In the exemplary switch 100′, n = m = k = 2. To simply the drawings, Figures 9(a) through 9(g) do not show the second input module (IM(1)), the second central module (CM(1)), or the second output module (OM(1)). As shown in Figure 9(a), to simplify the explanation, as well as to obtain desynchronization more

quickly, and in one embodiment of the output link arbitration method 230', the order of the VOQ(i,j,h) in IM(i) is redefined as VOQ(i,hk+j) as shown. Thus, in general, a pointer for use with the outgoing link arbitration method 230' and following a round-robin discipline, will cycle through the VOQs as follows:

VOQ(i,0,0);
VOQ(i,1,0);
...;
VOQ(i,k-1,0);
VOQ(i,0,1);
VOQ(i,1,1);
...;
VOQ(i,k-1,1);
...;
VOQ(i,0,n-1);
VOQ(i,1,n-1);
...;
VOQ(i,k-1,n-1).

In this way, the arbiter will cycle through a particular VOQ within various groups of VOQ first, and then through subsequent VOQs within each group.

In the following example, assume that VOQ(0,0), VOQ(0,3), VOQ(0,4), and VOQ(0,6) are non-empty. As shown in Figure 9(b), these non-empty VOQs each broadcast a request to all link arbiters in their input module (IM(0)). (Recall, e.g., the first communication 215 of Figure 2, as well as 310 and 320 of Figure 3.)

As shown in Figure 9(c), it is assumed that arbiters associated with outgoing links $L_i(0,0)$, $L_i(0,1)$ and $L_i(0,2)$ prefer VOQ(0,0), (VOQ(0,0) and VOQ(0,1), respectively. Since VOQ(0,0) is among those VOQs to broadcast a request, the arbiters associated with outgoing links $L_i(0,0)$ and $L_i(0,1)$ each send a grant signal back to VOQ(0,0). On the other hand, since VOQ(0,1) was empty and did not broadcast a request, the arbiter associated with outgoing link $L_i(0,2)$ will try subsequent VOQs until one that sent a request (i.e., a non-empty VOQ) is encountered. In this case, the next VOQ that sent a request is VOQ(0,3). Accordingly, as shown in Figure 9(c), the arbiter associated with outgoing link $L_i(0,2)$ sends a grant signal back to VOQ(0,3). (Recall, e.g., the second communication 237 of Figure 2, as well as the method 230′ of Figure 4.)

Referring to both Figures 9(c) and 9(d), it is assumed that arbiters associated with virtual output queues VOQ(0,0) and VOQ(0,3) both prefer outgoing link $L_i(0,0)$. Since both outgoing link $L_i(0,0)$ and $L_i(0,1)$ broadcast a grant to VOQ(0,0), the arbiter associated with VOQ(0,0) must select one. In this example, it selects outgoing link $L_i(0,0)$ and sends a grant signal back as shown in Figure 9(d). On the other hand, since only the outgoing link $L_i(0,2)$ sent a grant to VOQ(0,3), the arbiter associated with VOQ(0,3) will try subsequent outgoing links until one that sent a grant is encountered. In this case, the next (and indeed the only) outgoing link to send a grant is $L_i(0,2)$. Accordingly, as shown in Figure 9(d), the arbiter associated with VOQ(0,3) sends a grant signal back to outgoing link $L_i(0,2)$. (Recall, e.g., the third communication 247 of Figure 2, as well as 510, 540 and 550

of Figure 5.)

Assuming that more iterations of the foregoing steps are possible within one cell time slot (or more in certain embodiments), notice that VOQ(0,4) and VOQ(0,6) are non-empty, but were not matched with an outgoing link during the first iteration. Accordingly, as shown in Figure 9(e), these VOQs may rebroadcast their requests. (Recall, e.g., 520 and 530 of Figure 5.) Note that if the outgoing link arbiters maintain a certain information, such a rebroadcast would not be necessary. The outgoing link $L_i(0,1)$ is the only non-committed link. Based on its pointer, the arbiter associated with the outgoing link $L_i(0,1)$ will prefer VOQ(0,0). However, since VOQ(0,0) was already matched with an outgoing link in the previous iteration, and therefore did not rebroadcast a request, the arbiter associated with outgoing link $L_i(0,1)$ will try subsequent VOQs until one that rebroadcast a request is encountered. In this case, the next VOQ that rebroadcast a request is VOQ(0,4). As shown in Figure 9(f), the arbiter associated with the outgoing link $L_i(0,1)$ sends a grant back to VOQ(0,4). Finally, as shown in Figure 9(g), the arbiter associated with VOQ(0,4) sends a grant back to the outgoing link $L_i(0,1)$.

Although not shown in Figures 9(a) through 9(g), each outgoing link that was matched with a non-empty VOQ during the first phase of the exemplary cell dispatch scheduling invention will request a central module. Arbiters at the central modules will arbitrate such requests. Once an outgoing link L_o of a central module is matched with an outgoing link L_i of an input module (and

therefore to a VOQ), the state information (e.g., pointers) for each of the arbiters may be updated and the cells may be dispatched from those VOQs matched with an outgoing link L_o.

§ 4.6 CONCLUSIONS

The disclosed cell dispatch scheduling invention does not require buffers in the second stage, thereby advantageously avoiding cell out-of-sequence problems. Further, unlike random dispatching schemes having limited (e.g., 75%) throughput unless internal bandwidth is expanded (e.g., speed-up ≈ 1.582 is applied), simulations have shown that the cell dispatch scheduling invention can achieve 100 % throughput (independent of the number of iterations of the first phase) under uniform traffic. More specifically, a VOQ that fails to win contention has to store backlogged cells. Under uniform traffic, every VOQ keeps such backlogged cells until the idle state (i.e., the state in which the internal link is not fully utilized) is eliminated -- that is, until the stable state (See, e.g., the article, N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," IEEE Trans. on Communications, Vol. 47, No. 8, pp. 1260-1267 (Aug. 1999). This article is incorporated herein by reference) is reached. Once in the stable state, every VOQ is occupied with backlogged cells. In such a state, the arbiters (e.g., round-robin pointers) become desynchronized. Consequently, even when the offered traffic load is full, no contention occurs in the stable state. Therefore, the dispatch invention achieves 100 % throughput under uniform traffic.

Such 100 % throughput under uniform traffic occurs independent of the number of iterations. However, as the number of iterations (e.g., possible per cell time slot) increases, the delay performance becomes better. This is because the matching between VOQs and outgoing links L_i within an input module (IM) will improve. When the offered traffic is low, the desynchronization of the arbiters (e.g., round-robin pointers) is less likely achieved. In such a case, using less iterations negatively impacts performance. This is because the matching between VOQs and outgoing links L_i within an input module (IM) will worsen.

Even under bursty traffic, the dispatch invention provides 100 % throughput, which is also independent of the number of iterations of the first stage. However, the delay performance of the bursty traffic is worse than that of more uniform traffic under heavy load conditions.

The throughput of the cell dispatch scheduling invention is better than that of random dispatching even when traffic is unbalanced (i.e., not uniform). Further, the fact that the cell dispatch scheduling invention can use round robin arbiters ensures fairness under non-uniform traffic conditions.

The cell dispatch scheduling invention has a time complexity O(log nk) for each iteration of the first phase. Therefore, if there are m iterations (such that outgoing links are matched with the VOQs in an IM), the time complexity of the first phase is O(m log nk). The second

phase has a time complexity of O(log k). Therefore, the time complexity of the cell dispatch scheduling invention is approximately O(m log nk) = O(m log N), where N is the number of ports. If the number of iterations of the first phase is set to i, where 1 ≤ i ≤ m, the time complexity is expressed as O(i log N). Given this time complexity, the cell dispatch scheduling invention is scalable and may be used successfully in large scale switches.